Quality-Control Algorithm for Adaptive Streaming Services Over Wireless Channels

Sergio García, Julián Cabrera, and Narciso García

Abstract—**Dynamic Adaptive Streaming over HTTP (DASH) is a recent MPEG standard for IP video delivery whose aim is the convergence of existing adaptive-streaming proprietary solutions. However, it does not impose any adaptation logic for selecting the quality of the media segments requested by the client, which is crucial to cope effectively with bandwidth fluctuations, notably in wireless channels. We therefore propose a solution to this control problem through Stochastic Dynamic Programming (SDP). This approach requires a probabilistic characterization of the system, as well as the definition of a cost function that the control strategy aims to minimize. This cost function is designed taking into account factors that may influence the quality perceived by the users. Unlike previous works, which compute control policies online by learning from experience, our algorithm solves the control problem offline, leading promptly to better results. In addition, we compared our algorithm to others during a streaming simulation and we analyzed the objective results by means of a Quality of Experience (QoE) oriented metric. Moreover, we conducted subjective tests to complete the evaluation of the performance of our algorithm. The results show that our proposal outperforms the other approaches in terms of both the QoE-oriented metric and the subjective evaluation.**

Index Terms—**Adaptive streaming, DASH, quality of experience, stochastic dynamic programming.**

I. Introduction

VIDEO Traffic has definitely become the most significant part of Internet traffic and will continue increasing over the next years [2]. The ongoing increase of the average download speed, estimated as a 17% in the period 2012–2013 [3], should entail a better user satisfaction in video streaming services. However, given the current heterogeneous range of end-devices and Internet access connections, this is not always happening. This discrepancy is especially significant in the case of wireless channels, where phenomena like dead spots or fading entail a time-varying fluctuation of the channel's available throughput that has negative impact on the users' experience. In addition, since traffic from wired devices will be overcome by traffic from wireless and mobile terminals in the short term [2], this behavior must be successfully tackled in a transparent way for users, who generally are not aware of the underlying mechanisms in a multimedia transmission and mainly expect a smooth playback at a reasonable quality.

Traditional push-based streaming approaches like Real-time Transport Protocol (RTP) imply that the content server takes the decisions on which quality level is more appropriate regarding the current state of the system, given that it gets feedback from the client through RTP Control Protocol (RTCP). In contrast, pull-based methods rely on the client, who is responsible for requesting content based on its own measures, as long as bitrate adaptation is provided [4]. This allows the use of streaming over HTTP techniques which highly reduce the complexity of the servers and support the implementation of different adaptation algorithms. These solutions can therefore help achieve a higher QoE in video streaming services since the adaptation is performed bearing the specific characteristics of every client and every channel, instead of using a generic server-side algorithm.

Several proprietary implementations of adaptive streaming are currently being used in video transmission, including HTTP Live Streaming (HLS) or Smooth Streaming. However, the disparity of solutions and media formats has recently promoted the definition of the DASH standard by MPEG [5], whose objective is to provide a specification that leads to interoperability between servers and clients vendor-independent. The scope of the standard does not describe the full adaptive-streaming service, but instead describes the format of the manifest files and the segments. The adaptation logic therefore remains as a configurable feature and research is being done to develop an algorithm capable of enhancing user perception. Moreover, the current proprietary solutions can be improved as their performance under different channel models is not yet flawless. According to previous studies [6], the evaluated commercial players are either too aggressive or too conservative when reacting to the observed changes, and hence their performance can be improved.

In this paper, we propose an adaptation algorithm where the segment requests are ruled by policies which map a control parameter (the quality level, related to a certain video bitrate) to every possible state of the system (that is, the set of monitored variables). First the system is modeled as a discrete-time stochastic dynamic system and then the optimal policies are calculated offline using Stochastic Dynamic Programming (SDP) techniques. These policies are optimal in the sense that they minimize the long-term average of a certain cost function, which is also proposed in this paper to reward and penalize situations

that may result in high and low QoE, respectively. The SDP approach is highly convenient due to the nature of the problem and has already been followed in previous works for video wireless transmission strategies [7] and Quality of Service (QoS) adaptation control [8].

We also include an evaluation of the proposed algorithm by comparing its performance to other adaptation algorithms for both DASH and HLS solutions. After conducting a streaming simulation, we study objective parameters related to the QoE and try to estimate their impact on the users' perception by means of existing metrics. Moreover, we also conduct subjective experiments to compare these algorithms and complete the analysis of their performance.

The paper is organized as follows. Section II describes some of the existing algorithms in the literature and justifies the need for an optimal and mathematically accurate one. Section III presents an overview of the system and describes the control problem, whereas Section IV characterizes the models involved in it. Section V describes the formulation of the problem, including the proposed cost function. Then, Section VI presents the simulations framework and the comparison among algorithms, including subjective tests. Finally, in Section VII we include the conclusions of the work.

II. Related Work

Previous research has been done on adaptation algorithms, either through heuristic or mathematical approaches. Liu [9] proposes an empirical logic that estimates the available bandwidth from the previous download time and requests a tolerable bitrate. This model is overly simplistic and thus some studies, as the presented by Miller [10], include monitoring also the receiver's buffer level to avoid underflows. However, these models do not take advantage of a probabilistic model of the system and do not either consider the users' QoE.

Mathematical approaches have also been tackled, either with SDP methods [11], [12], which require a probabilistic characterization of the problem, or via reinforcement learning techniques [13], [14], where a model of the system is dynamically built and enhanced in execution time. Though this last solution is appropriate to adapt the client to the specific conditions of the network and the channel, the results show that the convergence of the intelligent agent may take up to thirty minutes. Besides, during the first minutes of playback the quality is especially poor, as the agent still does not have relevant data. This behavior is unbearable in a streaming context, since the average duration of online video content has been estimated as 5.2 minutes in recent researches [15].

We have consequently focused on studying the existing approaches that apply SDP techniques and how they formulate the optimization problem. Andelin [11] studies the quality adaptation for content encoded with Scalable Video Coding (SVC). Though it could be a reasonable encoding scheme and is supported by the standard, the majority of the video content available on the Internet is instead encoded with Advanced Video Coding (AVC). Developing an adaptation algorithm for SVC, although it could be more efficient, would involve a re-encoding of existing content and would make it impossible to compare our solution to other protocols like HLS, which use AVC. Besides,

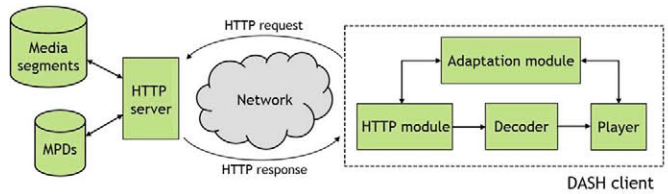

Fig. 1. Overview of a DASH system.

Kalva [16] has performed a comparison between the storage and bandwidth costs of a DASH server using either AVC or SVC, and they have determined that if the number of streaming sessions is relatively high, SVC expenses overtake the costs incurred with AVC. For these reasons we design our system with an AVC approach, which is more generic and extendable nowadays.

Regarding Xing's work [12], they do use SDP optimization for AVC content, but they address the specific issue of streaming through several wireless connections at the same time, taking advantage of the 3G and the Wi-Fi modules of most mobile devices. Though they also consider the cost function in terms of QoE, they do not perform any subjective validation on the resulting policies and their model includes as many as eight system variables, which increase significantly the complexity of the control problem and hence the convergence time of the algorithm. This may be important if online computation should be needed, for instance to recompute the policies if the models change.

Apart from this, many studies focused on estimating the QoE in adaptive streaming systems have been presented in the recent years. For instance, Singh [17] proposes a metric that estimates the QoE considering video freezes and quantization parameters, and Claeys [14] involves also quality switches into the model. However, they both rely on objective parameters instead of evaluating the subjective perception with real users of the system. There are also works that address this issue, as the one developed by Tavakoli [18], which compares different adaptation strategies by means of subjective tests. Nevertheless, they do not evaluate or compare real adaptation algorithms, but they analyze the impact on the perceived quality of the adaptation alternatives.

III. System Description

In this section, we present an overview of the DASH system and outline the associated control problem that is detailed analytically in Section V. A functional block diagram of a simple DASH system is presented in Fig. 1. It involves two entities: the HTTP server, which holds the video segments and the manifest files, and the client, responsible for requesting the content and presenting it to the user.

When a client decides to stream a specific content from the server, the player first requests the manifest file listing the available qualities and segments. Then it has to sequentially request the video chunks, taking for each of them a decision on the most adequate quality according to the environment observed at that moment. It can even decide not to download any segment and delay the following request if this turns to be more beneficial. This may be the case, for instance, to prevent a buffer overflow or so as not to react promptly to short-term variations. In the

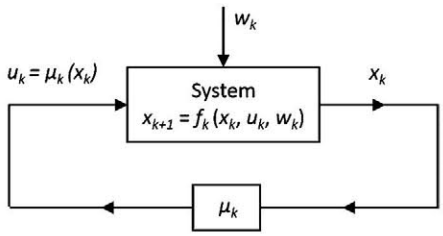

Fig. 2. Stochastic system dynamics. x_k denotes the system state at stage k; according to x_k, the controller takes action u_k; the system evolves to the next stage moving to state x_{k+1} as a consequence of the previous state, x_k, the action taken, u_k, and the influence of stochastic factor w_k.

meanwhile, the downloaded chunks will be decoded and presented to the user. Therefore, the client's buffer will be gradually filled with the downloaded segments and periodically drained due to the playback.

In order to apply SDP, our problem must be defined as a discrete-time stochastic system that evolves temporally in stages, like the one presented in Fig. 2 [19]. At each stage k, the system is in a state x_k (which includes all the information gathered from the environment) and the controller applies the control action u_k (in our case, the quality to be requested) according to the value of x_k. This selection, along with state x_k and stochastic disturbance ω_k, determine the evolution of the system state towards stage $k+1$, i.e., x_{k+1}, through function f_k.

Due to the discrete nature of the system, it can be characterized in terms of the transition probabilities among states, $P(x_{k+1}|x_k, u_k, \omega_k)$, which can be derived from function f_k and the probability distribution of ω_k. Besides, if the action selected by the controller at every stage incurs a certain cost $g_k(x_k, u_k, \omega_k)$, and the number of stages is high, we can formulate an infinite horizon problem and minimize the average cost per stage, defined as

$$\lim_{N\to\infty} \frac{1}{N} E\left\{ \sum_{k=0}^{N-1} g_k(x_k, u_k, \omega_k) \right\}. \quad (1)$$

The aim of SDP is hence finding a control policy μ_k, namely a mapping between the system states x_k and the control parameters u_k, which is optimal in the sense of minimizing (1) given $u_k = \mu_k(x_k)$. We will also focus on finding a stationary policy, that is, with the same control selection rule at every stage, denoted simply as μ.

This method allows us to compute offline the optimal control policies, given a probabilistic model of the DASH system. Consequently, a streaming client only needs to store this control law and select the quality of every segment as the output of function μ applied to the current state of the system.

The elements involved in the system are formulated mathematically in the following sections. Specifically, Section IV describes the environment variables and their models, whereas Section V integrates these definitions to formalize the evolution of the whole stochastic system.

IV. System Modeling

With the information extracted from Section III, the state variables that make up system vector x_k can be identified and modeled subsequently. In particular, we have considered the variables b_k, which represents the client's buffer level at the beginning of stage k; bw_k, which stands for the average channel bandwidth during the download of the previous segment; and q_k, which holds the quality level requested for the previous segment.

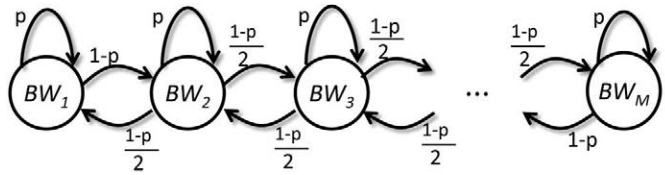

Fig. 3. Markov-chain model for the available bandwidth.

As for the control variable, u_k, it represents the quality to be requested for segment k. We have also included the option of delaying the following request for a certain amount of time, which will be noted as $u_k = 0$. Consequently, provided that there are R quality levels available at the server, the set of possible values for u_k is $\{0, Q_1, \ldots, Q_R\}$. In addition, we will denote the media bitrate associated to every quality level as $b_r(u_k)$ throughout this paper, measured in bits per second.

A detailed description of the system state models is presented afterwards, whereas the system dynamics is provided in Section V.

A. Channel Model

In contrast to traditional video transmission approaches, we are not interested in low-level characteristics like the packet loss probability, since the use of the reliable transport protocol TCP ensures that the media content is received without errors. However, this results in the download time fluctuating due to retransmissions or TCP's congestion control. In addition, wireless channels entail other issues, such as fading, that affect the download time of a segment. We must hence model the available bandwidth or throughput provided by the channel, for which the state variable bw_k is used.

Considering a wireless channel affected by Rayleigh fading, a first-order discrete-time Markov chain can be used to model the changes in the received signal to noise ratio [20]. Accordingly, the available throughput can also be modeled by means of a Markov chain where the states represent the different bandwidth values [11], [12].

Fig. 3 illustrates a generalization of the Markov chain proposed in [11] for an HTTP channel. In this case, bandwidth is quantified into M different values $\{BW_1, \ldots, BW_M\}$ and transitions are only possible to adjacent states, with equal probabilities. It shows that bandwidth tends to remain constant over a certain period of time and change smoothly, that is, from one state to its neighbors, which seems suitable for a wireless channel. Notwithstanding, such models can be adapted to the conditions of a specific channel by studying its behavior and obtaining its associated Markov chain [21].

In our work, we adopt the model proposed in [11]. The value of M can be adjusted considering that a high number will generate smaller quantification intervals and hence provide more accurate results, but it will also entail a larger variable space and higher computation times for the optimization. In addition, the state-remaining probability p may be selected as well in order

to model different channel behaviors: for instance, values close to one can represent low-interference channels, whereas low values imply frequent throughput fluctuations that may be due to high cross-traffic.

B. Segment Model

Regarding the video segments requested by the client, the only parameter influencing the evolution of the system is their size, i.e., their number of bytes. Greater segments will entail a higher download time than smaller chunks, and therefore the client's buffer filling rate will vary as segment size does.

We have considered DASH segments with constant duration (denoted by T, in seconds). Segment size is hence directly proportional to the media bitrate and is given by the expression $s_k = T \cdot br\,(u_k)$. In the case of using Constant BitRate (CBR) encoding, size keeps also constant, but this technique is not useful since the resulting quality may vary over time and degrade in complex video scenes. We are therefore interested in content encoded with Variable BitRate (VBR) schemes, which ensure a stable quality but also involve a slight deviation in the segment size that must be contemplated.

By observing the histograms of segment sizes encoded with VBR, we can assume that segment size can be modeled through a discrete Gaussian distribution whose mean is $T \cdot br\,(u_k)$, whereas its standard deviation depends on the complexity of the scene. Therefore, distinct control actions imply downloading segments with different sizes, which influence differently the evolution of the system.

V. Problem Formulation

A. System Dynamics

Once we have described the channel and segment size models, we can analyze the whole system dynamics in order to calculate the state transition probabilities and compute the control policies.

The buffer level, b_k, represents the number of segments available in the receiver's buffer. It can take integer values from 0 to B_{MAX}, which holds the maximum number of segments that a client may store and depends on the end-device characteristics. As outlined in Section III, the buffer level rises when a segment is downloaded and descends when segments are played. Therefore its evolution to the following stage b_{k+1} depends on the previous buffer occupancy b_k, the available throughput bw_k and the size of the downloaded segment s_k, which is influenced by the requested quality u_k.

In particular, for the cases when a segment is requested $(u_k \neq 0)$, at the beginning of the next stage there is always one more segment in the buffer. However, the number of extracted chunks depends on the elapsed time Δt_k divided by T to take into account the segment duration, and rounding to the nearest integer since only full segments can be removed from the buffer. Noticing that the elapsed time depends on the segment size and the available throughput, we can therefore define

$$b_{k+1} = b_k + 1 - \left[\frac{\Delta t_k}{T}\right] = b_k + 1 - \left[\frac{1}{T}\frac{s_k}{bw_k}\right]. \quad (2)$$

We must also take into account that for low throughput values, the elapsed time may be relatively high and therefore the previous expression could be lower than zero. However, as this has no physical meaning, we must set a lower bound to b_{k+1}. Since at the beginning of a new stage there is always at least one segment in the buffer, the one that has just been downloaded, we can modify (2) into (3).

$$b_{k+1} = \max\left\{b_k + 1 - \left[\frac{1}{T}\frac{s_k}{bw_k}\right], 1\right\}, u_k \neq 0 \quad (3)$$

The case of delaying the following request $(u_k = 0)$ is easy to derive from the previous reasoning. In our formulation, we define the delay time as a parameter, T_{delay}. It represents the time during the client waits idle before requesting a new segment. The number of segments removed from the buffer during that time depends on the elapsed time as well, but in this case the minimum buffer level is zero, as we don't add new segments and the buffer may end empty in the next stage. System equation in this case is given by expression (4).

$$b_{k+1} = \max\left\{b_k - \left[\frac{T_{delay}}{T}\right], 0\right\}, u_k = 0 \quad (4)$$

Regarding the remaining system variable, q_k, it represents the value of the last requested quality. Consequently, when a segment is requested, its value at stage $k+1$ is equal to the control parameter $q_{k+1} = u_k$; whereas for the delay case, it takes the same value as in the previous stage, that is, $q_{k+1} = q_k$. Though the evolution of this variable does not influence the system dynamics, it has been considered into the system state due to its importance in the cost function design to take into account the possible QoE effects of switching from one quality to another between consecutive segments. Its behavior is given by system (5).

$$q_{k+1} = \begin{cases} u_k, u_k \neq 0 \\ q_k, u_k = 0 \end{cases} \quad (5)$$

B. Transition Probabilities

The set of transition probabilities between states is used to characterize the evolution of the system. Based on the selected action u_k, the transition from present state x_k to next state x_{k+1} will be given by

$$P\{x_{k+1}|x_k, u_k, \omega_k\} = P\{b_{k+1}, bw_{k+1}, q_{k+1}|b_k, bw_k, q_k, u_k, s_k\}. \quad (6)$$

We must note that the state of the system is given by the vector $x_k = (b_k, bw_k, q_k)$ and that segment size s_k influences the probabilities indirectly through the control action u_k. However, it does not take part in the state of the system since the value of s_k does not depend on its previous value s_{k-1}.

Taking into account that system variables b_{k+1}, bw_{k+1} and q_{k+1} are independent of each other and knowing how they evolve (either through a system equation or with a Markov model), expression (6) can be rewritten as

$$P\{b_{k+1}|b_k, bw_k, u_k, s_k\} \cdot P\{bw_{k+1}|bw_k\} \cdot P\{q_{k+1}|q_k, u_k\}. \quad (7)$$

The central term of this expression is obtained through the Markov chain model presented in Section IV, whereas the right and left terms can be computed using the system (3), (4) and (5), and the probability distribution of s_k. Finally, we can build the transition probability matrix, which holds the probability of transitioning from one state to another due to selecting every control action.

C. Cost Function

In order to complete the formulation as an SDP problem, we have to define a measure of the cost g_k which the system incurs when a control action is selected from a certain state. As presented in Section I, this function is designed to penalize the situations that may cause a decrease in the users' QoE. To achieve this target we have focused on reacting to three objective parameters that influence subjective perception [14]: quality level, quality switches and video freezes, which are represented by the cost components detailed below.

The proposed cost function for the case of not delaying the request $(u_k \neq 0)$ is defined by (8).

$$g_k(x_k, u_k \neq 0) = \begin{cases} d'_k + \beta(q_k - u_k) + \gamma \cdot c_k, & d'_k \geq 0 \\ \alpha \cdot \left(1 - e^{d'_k}\right) + \beta(q_k - u_k) + \gamma \cdot c_k, & d'_k < 0 \end{cases} \quad (8)$$

This cost function favors the selected bitrate to be close to the available bandwidth. It also distinguishes whether the selected bitrate is higher or lower than the last known throughput value. In the first case, there is a risk of causing a buffer underflow, so g_k will penalize it more, as will be shown. This different behavior is controlled according to the value of d'_k. This variable represents a modulated bitrate difference between the throughput and the requested bitrate.

We define the bitrate difference between the network throughput and the media bitrate of the requested quality as

$$d_k = bw_k - br(u_k). \quad (9)$$

In order to modulate this difference, the buffer level should also be considered. For instance, a low buffer level may involve requesting lower qualities, in order to avoid underflow and favor a buffer level increase. To capture this behavior we define a target buffer level B_{opt} that represents a desired value that the client should try to keep during the streaming session, and modulate the previous bitrate difference defined in (9) as

$$d'_k = bw_k \frac{1 + b_k/B_{opt}}{2} - br(u_k). \quad (10)$$

Inspecting this expression, we can see that when the buffer level is B_{opt}, the behavior is the same as in (9). Otherwise, when the buffer level is under B_{opt}, the bitrate is compared with a value which is lower than the throughput, and hence lower bitrates will be preferred to help increasing the buffer level. Similarly, if b_k is greater than B_{opt}, selecting higher bitrates is favored. An appropriate value for this parameter should consider the implementation characteristics of the adaptive streaming system and it will be defined in Section VI for our simulation set up.

The cases where $d'_k > 0$ are those where the throughput is higher than the requested bitrate. This situation is preferred to the defined by $d'_k < 0$, since it means that the selected bitrate is more likely to be tolerable by the network and may not cause a buffer level decrease. We therefore propose using a linear function for the interval $d'_k > 0$ and a negative exponential one for $d'_k < 0$, since for lower values of $|d'_k|$, the case where d'_k is negative will entail a higher cost.

In order to reduce the quality switches, we also include in the cost function the difference between the last requested quality and the selected one, $q_k - u_k$, to penalize a possible QoE reduction due to high quality variations.

Besides, if we want to prevent video freezes, the buffer level must be controlled to avoid running out of segments in the client. This is translated into the cost model by means of c_k. It represents a parabolic function which favors keeping b_k close to B_{opt} and penalizing especially low buffer values, as is shown in (11).

$$c_k = \left(\frac{b_k}{B_{opt}}\right)^2 - 2 \cdot \frac{b_k}{B_{opt}} + 1 \quad (11)$$

These three cost components are added together into a single expression, where parameters α, β and γ are used to weight them properly. As happened with B_{opt}, the value of this parameters must be chosen based on the specific implementation characteristics.

Finally, the case $u_k = 0$ must be designed to be only selected when both the available bandwidth and, specially, the buffer level are both relatively high. This is adopted to avoid leading to a buffer underflow if it did not have enough segments or if during the delay time the channel throughput has a strong decrease. Taking this behavior into account, we penalize this selection for states with the buffer level far from B_{MAX} and the estimated bandwidth close to the minimum value, BW_1. Weighting properly each term with parameters δ and ε as we did in (8) to make comparable the different addends, we can derive the total cost function as

$$g_k(x_k, u_k = 0) = \delta \cdot \left\{ \left(\frac{b_k}{B_{MAX}}\right)^2 - 2 \cdot \frac{b_k}{B_{MAX}} + 1 \right\} + \varepsilon \cdot \frac{BW_1}{bw_k}. \quad (12)$$

VI. Simulations and Results

We performed a streaming simulation with our control-policies algorithm, another DASH logic based on monitoring the buffer level [10] and the HLS implementation currently used by a commercial player. We then compared the results objectively, especially in terms of average segment quality, video freezes and quality switches; and subjectively, by means of assessment tests showing some portions of the resulting sequences to several users.

A. Experiments Setup

We first encoded a two-hour-long HD video movie in 14 quality streams with average bitrates Q_i distributed between 0.1 and 4.5 Mbps following a similar scheme to the distributed DASH dataset [22]. Each stream was segmented in chunks with a constant duration of 2 seconds (therefore, $T = 2$) and

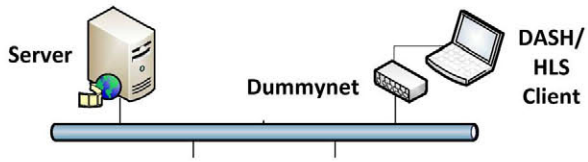

Fig. 4. Network topology used for the simulations.

they were stored in a HTTP server connected to our computer through a LAN, as illustrated in Fig. 4.

In order to dynamically limit the available bandwidth and simulate the channel behavior, we used the open source tool DummyNet [23]. We instantiated a sequence of throughput changes that followed the Markov chain model presented in Section IV for a wireless channel, with $M = 21$ states and a remaining probability $p = 0.8$. The quantification values were chosen as $BW_1 = 100$ kbps and $BW_i = 0.25 \cdot (i-1)$ kbps for $i = 2, \ldots, 21$. The transitions between states of the chain were performed every second, and the same throughput evolution was repeated in all the simulations.

Regarding the configurable parameters of our model, we have chosen the maximum buffer level following [14] as $B_{MAX} = 10$ segments, which means that the client is capable of storing up to 20 seconds of video. This value represents in the scenario we are considering a medium size buffer that allows both: (i) to absorb channel fluctuations up to some extent by its size (thus ensuring a reasonable level of performance of the algorithms considered); (ii) to exploit the *a priori* knowledge of the system behavior, especially the channel, used in our approach. As for the optimum buffer level, according to the experiments we have carried out, we have determined its value as $B_{opt} = 0.75 \cdot B_{MAX} = 7$, since it seems a reasonable target to keep during the streaming session. Besides, we have chosen a delay time that is equal to the segment duration, that is, $T_{delay} = 2$, to ensure that only one segment can be removed from the buffer when this decision is taken.

For the parameters involved in the cost function design, we have performed the computation of the control policies for several combinations of values $(\alpha, \beta, \gamma, \delta, \varepsilon)$. We have analyzed the resulting policies and those whose actions did not follow the criteria used to design the cost function (such as requesting: similar bitrates for consecutive stages, higher quality segments as the buffer level and the throughput increase, or choosing the delay option when the buffer level and the throughput are high) were discarded. Consequently, the final values involved in the cost function are $\alpha = 0.5$, $\beta = 7$, $\gamma = 4.4$, $\delta = 100$ and $\varepsilon = 100$.

Finally, in the case of the segment size model, we used the segment sizes existing in the DASH dataset [22] to perform a Maximum Likelihood Estimation (MLE) and obtained the mean and the standard deviation for every quality u_k.

After the policies were calculated, we used a DASH client implemented on top of a sample player distributed with the open source library libdash [23], in which we added the adaptation algorithms and the necessary modules to register the relevant parameters (such as the selected bitrate or the buffer level) of the streaming session. For the HLS simulation we used an existing commercial video player whose adaptation logic tries to download the maximum quality available that can be supported with the measured throughput.

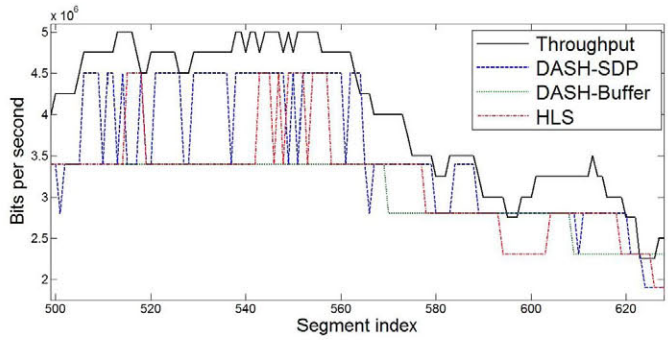

Fig. 5. Evolution of the throughput and the requested bitrates for the three evaluated algorithms during a portion of the simulation.

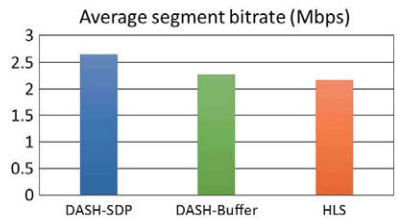

Fig. 6. Average segment bitrate obtained during the simulations for the three algorithms under consideration.

B. Objective Results

An extract of the quality evolution during the simulations is presented in Fig. 5, measured in bits per second of every selected segment. The progression of the available throughput for that period of time is also shown. The illustrated period represents an interval of 128 segments, that is, 254 seconds.

This figure shows that our algorithm manages to request on average higher bitrates than the others, though this is achieved by producing more quality switches. Although our algorithm tried to reduce the number of these variations through the cost function, we must note that they mostly happen between consecutive quality levels. On the other hand, the HLS algorithm involves several quality switches as well, but it tends to request lower bitrates on average. The case of the buffer level algorithm is the opposite, since it has few quality fluctuations, but this leads to a generally lower quality level.

We now provide some objective results that have been calculated with the data registered throughout the three simulations, and will later try to study their impact in the QoE. First of all, we have computed the average segment bitrate, which is presented in Fig. 6. As can be observed, the value achieved with the SDP algorithm is greater than the others, which confirms the behavior outlined in Fig. 5.

Another monitored variable is the length of the playback freezes that happened during the streaming, which usually influence negatively the users' QoE. From these values we derived the average freeze length and the average freeze frequency, that is, the number of freezes per unit of time. The results are presented in Fig. 7, which shows that the proposed algorithm achieves the smallest average duration, much lower than the value of HLS. However, the frequency of the freezes is greater for our SDP algorithm, which may be due to the fact that this particular implementation of HLS, unlike DASH, does not resume immediately the playback when receiving a new segment, but instead waits until the buffer reaches a

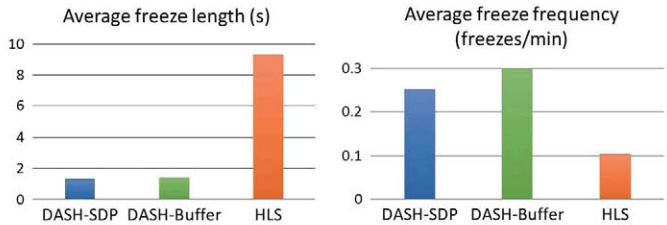

Fig. 7. Average freeze length and frequency obtained during the simulations for the three algorithms under consideration.

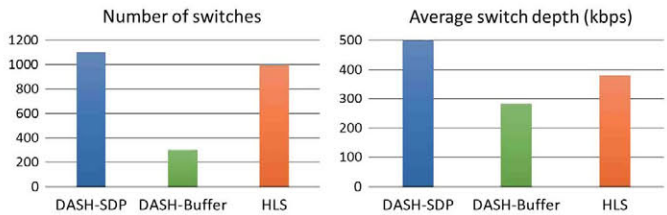

Fig. 8. Number of quality switches and average switch depth obtained during the simulations for the three algorithms under consideration.

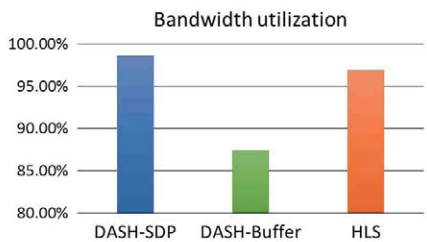

Fig. 9. Average bandwidth utilization obtained for the simulations.

certain threshold before starting to display segments again. This hence tends to increase the length of freezes and decrease their frequency, as can be extracted from the results.

Quality switches may also influence negatively the user's satisfaction [14], [18]. This effect grows higher as the number of switches increases, but the average switch depth, that is, the bitrate difference between two consecutive segment requests, is also relevant. Fig. 8 shows both parameters obtained for the three simulations and reveals, as expected, that our SDP-based algorithm entails the highest values. Nevertheless, in order to evaluate the real impact on the users, we have performed subjective evaluations of the algorithms (see Section VI-C). The obtained results will show that these quality variations across similar bitrates do not penalize the perceived quality, especially when switching between high bitrates.

Bandwidth utilization, defined as the ratio of the requested bitrate and the available throughput, averaged over every segment, has also been calculated. This measure is useful because it shows how well the algorithm has adapted to the network conditions, managing to use all the available bandwidth to download segments. Fig. 9 shows these results, pointing out that the three algorithms achieve a high utilization (above 85%), which means network resources are not being wasted during the streaming. Besides, SDP algorithm and HLS reach a value which is close to 100% because for several segments the client requests a bitrate which is higher than the available throughput.

We finally carried out a numerical comparison between the algorithms based on previously proposed QoE measurement methods through objective parameters. Claeys [14] proposes

TABLE I
COMPARISON OF QOE RESULTS FOR THE SIMULATIONS

Solution	Algorithm	QoE	Q (%)	F (%)	S (%)
DASH	SDP-based	3.093	60.95	8.26	5.9
DASH	Buffer-based [10]	2.650	54.94	10.2	0.6
HLS	Player's	2.923	52.37	1.6	2.5

modeling users' satisfaction as a function directly proportional to the average segment quality (factor $\boldsymbol{Q}$) penalized by the frequency and length of video freezes (factor $\boldsymbol{F}$) and quality switches (factor $\boldsymbol{S}$), which are defined as follows. In these expressions, $\boldsymbol{N}$ stands for the total number of segments; $\boldsymbol{Q_1}$ and $\boldsymbol{Q_R}$ are the minimum and maximum quality levels, respectively; $\boldsymbol{fr_{freq}}$ and $\boldsymbol{fr_{time}}$ are the average freeze frequency and freeze length; and $\boldsymbol{sw_{number}}$ and $\boldsymbol{sw_{depth}}$ represent the number of switches and their average bitrate depth.

$$Q = \frac{1}{N \cdot br(Q_R)} \sum_{k=0}^{N-1} br(u_k)$$

$$F = \frac{7}{8}\left(\frac{\ln(fr_{freq})}{6} + 1\right) + \frac{1}{8}\left(\frac{\min(fr_{time}, 15)}{15}\right)$$

$$S = \frac{sw_{number} \cdot sw_{depth}}{N \cdot [br(Q_R) - br(Q_1)]}$$

Claeys proposes modeling the global QoE by weighting properly the three previous factors so that users' preferences and aversions to the previous factors are considered. The subjective tests he performed lead to (13), which is the expression we used to evaluate objectively the overall results of our simulations. Table I shows the outcomes of applying this metric to our simulations for an instantiation of the channel model running for two hours. This instantiation has been used for the simulation of the three algorithms.

$$QoE = 4.85 \cdot Q - 4.95 \cdot F - 1.57 \cdot S + 0.5 \qquad (13)$$

As we expected from the previous results, our algorithm requests a higher average quality $\boldsymbol{Q}$, measured as a percentage of the highest quality available. However, as extracted from factor $\boldsymbol{S}$, it has also the highest quality switching rate. Regarding the influence of freezes $(\boldsymbol{F})$, our logic achieves an intermediate punctuation between the others.

Based on these results, the proposed algorithm outperforms both buffer-based and HLS logics. Our adaptation logic requests a higher average quality than the other solutions and manages to maintain an acceptable number of freezes so as not to disturb substantially the watching experience. Although the number of quality switches is higher, the effects of this parameter in the users' perception will now be assessed in Section VI-C by means of subjective evaluation tests.

C. Subjective Tests

With the simulation results we conducted some subjective tests in order to validate the performance of the proposed algorithm in relation to the others, and to observe the users' perception regarding the greater number of quality switches attained with our method. Before showing the outcome of these subjective experiments, we detail relevant aspects like the structure of

TABLE II
CHARACTERISTICS OF THE TEST SEQUENCES

Video sequence	Content	Channel	Format
Seq. 1	Movie containing segments with fast and slow motion	High throughput	1920x800p 24 fps
		Low throughput	940x400p 24 fps
Seq. 2	Animation sequence with smooth motion	High throughput	1920x800p 24 fps
		Low throughput	940x400p 24 fps

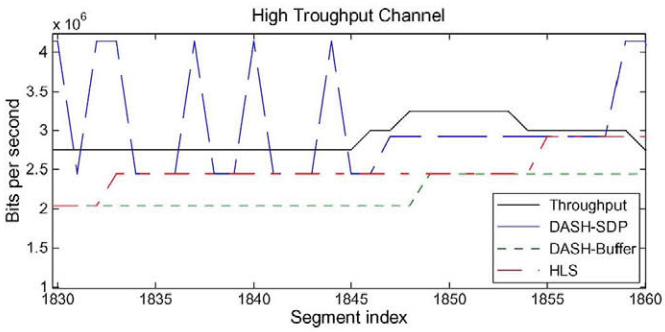

Fig. 10. Channel throughput and requested bitrates for the three algorithms during the High Throughput test sequences.

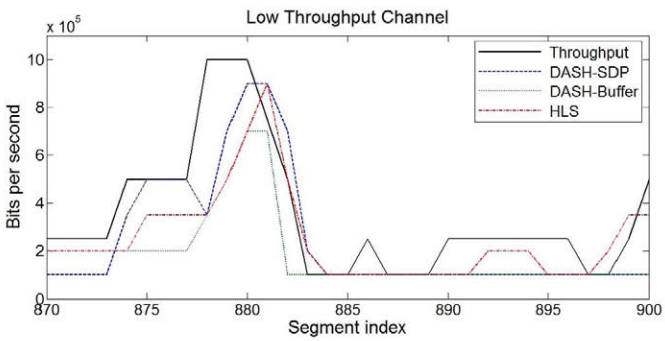

Fig. 11. Channel throughput and requested bitrates for the three algorithms during the Low Throughput test sequences.

the test material, the employed methodology and the observers' characteristics.

As for the test sequences, we extracted two clips from two commercial videos with different features, such as their type of content or their kind of motion. The clips were one-minute long (equivalent to thirty segments in our framework), since that length was enough to reflect the different behavior of the algorithms while keeping a reasonable duration for the users' evaluations. We then selected two portions of the previous streaming simulations, one of them where the channel presented a high average throughput, and the other when it had a lower throughput, in order to evaluate both scenarios. The main properties of the sequences are summarized in Table II.

Finally, we formed the four sequences presented in Table II using the series of segments requested by the three algorithms during the 'High Throughput' and 'Low Throughput' periods of time. This data are provided in Figs. 10 and 11 for both channel behaviors, along with the available throughput.

TABLE III
PAIRS OF SEQUENCES PRESENTED TO THE USERS

Display A	Display B
High Throughput SDP	High Throughput Buffer
High Throughput SDP	High Throughput HLS
Low Throughput SDP	Low Throughput Buffer
Low Throughput SDP	Low Throughput HLS

The evaluation methodology that has been used for the 'High Throughput' case is pair comparison [25], [26] with simultaneous presentation in two screens using a synchronized player. We chose this method because it is easier for the users to notice the difference between the test videos when they are presented together, due to their long length. Notwithstanding, we have also applied sequential pair comparison for the 'Low Throughput' set of videos, since they include video freezes which cause a synchronization loss between the sequences, making the simultaneous evaluation harder for the users.

After starting every assessment session with a brief explanation of the experiment to the observers, we showed them the different clips by pairs, in which one of them always corresponded to our algorithm, and the other could belong to the DASH buffer-based logic or to the HLS algorithm. The set of pairs shown to the users is presented in Table III, and was repeated for both contents (Seq. 1 and Seq. 2). Besides, the display of every pair of videos in screen A or B was randomized for each observer to avoid contextual effects.

After the presentation of each pair of videos, a message appeared on the screen over grey background during 5 seconds, allowing the observers to write in the questionnaire. They had to choose between one of the presentations of the pair, selecting the one they preferred. In addition, the tests were done individually, that is, one observer each time.

The subjective evaluations were carried out at the Universidad Politécnica de Madrid, in a lab with controlled ambient light. The displays used for presenting the sequences were two 22″ Samsung televisions, with resolution of 1920 × 1080 and aspect ratio 16:9.

Eighteen observers participated in the experiments, all of them having normal visual and stereoscopic acuity, and color vision. The ages of the subjects were ranged between 20 and 44 years old, with an average age of 25.

The scores provided by the test participants were collected and are shown below grouped according to the channel type. The results for the 'High Throughput' channel are displayed in Fig. 12. We can see that our algorithm outperforms the others, especially the buffer-based logic which had the minimum number of quality switches.

As for the 'Low Throughput' channel, we can observe in Fig. 13 that the observers' scores are also better for the SDP algorithm. However, it is true that the results are more balanced compared with the HLS sequences, which may be explained because our algorithm requests lower bitrate segments to avoid buffer underflows and hence video freezes.

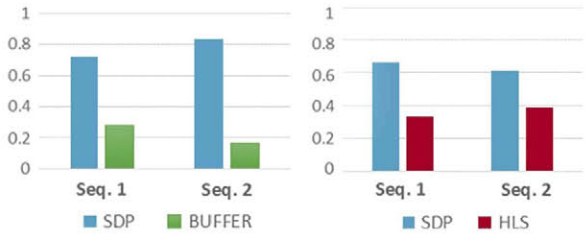

Fig. 12. Ratio of votes to total number of observers gathered for every pair of sequences belonging to the 'High Throughput' channel type.

Fig. 13. Ratio of votes to total number of observers gathered for every pair of sequences belonging to the 'Low Throughput' channel type.

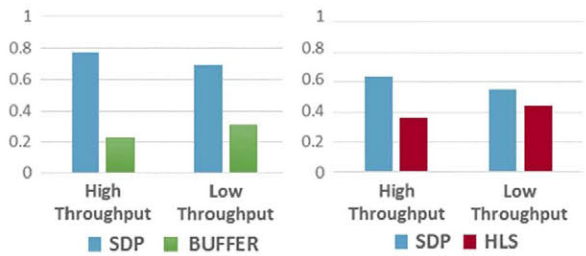

Fig. 14. Average ratio of votes to total number of observers for every channel.

We finally present in Fig. 14 the average results, independent of the video sequence, which compare the SDP algorithm with the other two approaches. We can firmly confirm that the performance of our solution is superior to the buffer-based logic in terms of QoE, since around three quarters of the observers preferred the video sequences generated with SDP. Regarding HLS, the results show that our algorithm also improves the subjective perception of the users, especially for high throughput situations.

VII. Conclusions

Transmission of on-demand audiovisual content via over-the-top (OTT) systems is currently gaining ground against classical streaming approaches, particularly in the case of delivering the content with adaptive streaming solutions. MPEG-DASH standard is still in early periods of adoption, but it starts to stand out as an extremely powerful adaptive-streaming technique, especially in the field of mobile devices. Transmission to this kind of clients is generally done through wireless networks, which suffer from a high throughput variability that tends to cause negative effects in the QoE.

We have therefore proposed an adaptation algorithm obtained by means of SDP, which relies on a probabilistic characterization of the system to compute offline the control policies mapping the environment information to the client requests. In this scenario we have designed a cost function that penalizes situations that may lead to a reduction of the QoE.

To evaluate our solution, we have performed streaming simulations using our algorithm, another DASH logic based mainly on monitoring the buffer level and a commercial implementation of HLS. The objective results show that the average quality requested with our algorithm is higher, but it also involves a relevant number of quality switches among segments. Regarding the video freezes issue, it attains an intermediate position between the other algorithms under consideration, leading to fewer playback interruptions than the buffer-based logic, but more than HLS, although their duration is shorter. We have also integrated all this information into an existing QoE-oriented metric, which has shown that globally our algorithms outperforms the others.

In order to validate these results, we have conducted subjective experiments to show the effects of the three considered algorithms in the perception of users, treating differently the cases of high throughput variations and of low throughput fluctuations. The outcomes of the tests confirmed that our SDP adaptation algorithm is around three times more appreciated by the observers than the buffer-based logic for both kinds of channels, which confirms the objective result. As for HLS, its results are closer to those of SDP but the performance of our algorithm was also considered better by the majority of the users, especially in the case of a channel with high throughput oscillations. Hence the quality variations among high bitrates present in our algorithm do not greatly penalize the subjective perception. Moreover, our solution achieves a better tradeoff between the requested video qualities and the resulting playback freezes, which is more noticeable for low-throughput channels.

Acknowledgment

We would like to thank Jesus Gutiérrez and Virginia Martín for their help during the conduction of the subjective tests.

References

[1] S. García, J. Cabrera, and N. García, "Quality-optimization algorithm based on stochastic dynamic programming for MPEG DASH video streaming," in *Proc. IEEE Int. Conf. Consumer Electron.*, Jan. 2014, pp. 588–589.

[2] Cisco Systems, Inc., "Networking index: Forecast and methodology, 2012–2017," Cisco Visual, May 2013.

[3] Akamai Technologies, Inc., "The State of the Internet. 1st quarter," 2013 Report vol. 6, no. 1.

[4] A. C. Begen, T. Akgul, and M. Baugher, "Watching video over the web: Part 1: Streaming protocols," *IEEE Internet Comput.*, vol. 15, no. 2, pp. 54–63, Mar.–Apr. 2011.

[5] "ISO/IEC 23009-1:2012—Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1; Media Presentation Description and Segment Formats".

[6] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. 2nd ACM Conf. Multimedia Syst.*, Feb. 2011, pp. 157–168.

[7] V. Miguel, J. Cabrera, F. Jaureguizar, and N. García, "Distribution of high-definition video in 802.11 wireless home networks," *IEEE Trans. Consumer Electron.*, vol. 57, no. 1, pp. 53–61, Feb. 2011.

[8] C. C. Wüst, L. Steffens, W. F. J. Verhaegh, R. J. Bril, and C. Hentschel, "QoS control strategies for high-quality video processing," in *Proc. 16th Euromicro Conf. Real-Time Systems*, Jun. 2004, pp. 3–12.

[9] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. 2nd ACM Conf. Multimedia Syst.*, Feb. 2011, pp. 169–174.

[10] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *Proc. IEEE 19th Int. Packet Video Workshop*, May 2012, pp. 173–178.

[11] T. Andelin, V. Chetty, D. Harbaugh, S. Warnick, and D. Zappala, "Quality selection for dynamic adaptive streaming over HTTP with scalable video coding," in *Proc. 3rd Multimedia Syst. Conf.*, Feb. 2012, pp. 149–154.

[12] M. Xing, S. Xiang, and L. Cai, "Rate adaptation strategy for video streaming over multiple wireless access networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2012, pp. 5745–5750.

[13] V. Menkovski and A. Liotta, "Intelligent control for adaptive video streaming," in *Proc. IEEE Int. Conf. Consumer Electron.*, Jan. 2013, pp. 127–128.
[14] M. Claeys, S. Latré, J. Farmaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming," in *Proc. 12th Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, May 2013.
[15] ComScore, Inc., August 2013 U.S. Online Video Rankings.
[16] H. Kalva, V. Adzic, and B. Furht, "Comparing MPEG AVC and SVC for adaptive HTTP streaming," in *Proc. IEEE Int. Conf. Consumer Electron.*, Jan. 2012, pp. 158–159.
[17] K. D. Singh, Y. Hadjadj-Aoul, and G. Rubino, "Quality of experience estimation for adaptive HTTP/TCP video streaming using H.264/AVC," in *Proc. IEEE Consumer Commun. Netw. Conf. (CCNC)*, Jan. 2012, pp. 127–131.
[18] S. Tavakoli, J. Gutiérrez, and N. García, "Subjective quality study of adaptive streaming of monoscopic and stereoscopic video," *IEEE J. Sel. Areas Commun. , Iss. Adaptive Media Streaming*, vol. 32, no. 4, pp. 684–692, Apr. 2014.
[19] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*, 1st ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1987.
[20] H. S. Wang and N. Moayeri, "Finite-state Markov channel - a useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, vol. 41, no. 1, pp. 163–171, Feb. 1995.
[21] D. Tsamis, T. Alpcan, J. P. Singh, and N. Bambos, "Dynamic resource modeling for heterogeneous wireless networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2009, pp. 1–6.
[22] S. Lederer, C. Mueller, C. Timmerer, C. Concolato, J. Le Feuvre, and K. Fliegel, "Distributed DASH dataset," in *Proc. MMSys*, Feb. 2013, pp. 131–135.
[23] M. Carbone and L. Rizzo, "Dummynet Revisited," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 2, pp. 12–20, 2010.
[24] C. Mueller, S. Lederer, J. Poecher, and C. Timmerer, "libdash - An open source software library for the MPEG-DASH standard," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2013.
[25] *Subjective Video Quality Assessment Methods for Multimedia Applications*, Rec. ITU-T P.910, ITU-T, 1999.
[26] J. Li, M. Barkowsky, and P. Le Callet, "Subjective assessment methodology for preference of experience in 3DTV," in *Proc. 11th IEEE IVMSP Workshop, Korea*, Jun. 2013, pp. 1–4.

Sergio García received the Telecommunication Engineering degree (five-year engineering program) from the Universidad Politécnica de Madrid (UPM), in 2013. He is currently working towards the Ph.D. degree in Telecommunication at UPM. Since 2013 he has been a member of the Grupo de Tratamiento de Imágenes (Image Processing Group) of the UPM. His research interests include adaptive streaming techniques and algorithms, as well as 3D graphics compression and rendering, especially in the field of point-cloud-based models.

Julián Cabrera received the Telecommunication Engineering degree and the Ph.D. degree in telecommunication, both from the Universidad Politécnica de Madrid (UPM), in 1996 and 2003, respectively. Since 1996 he has been a member of the Image Processing Group of the UPM. He was a Ph.D. scholar of the Information Technology and Telecommunication Programs of the Spanish National Research Plan from 1996 till 2001. Since 2001 he has been a member of the faculty of the UPM, and since 2003 he has been an Associate Professor of Signal Theory and Communications. His professional interests include image and video coding, design and development of multimedia communications systems, focusing on Multivew Video Coding (MVC), 3D Video Coding and video transmission over variable rate channels. He has been actively involved in European projects (Acts, Telematics, IST) and national projects.

Narciso García received the Ingeniero de Telecomunicación degree (five-year engineering program) in 1976 (Spanish National Graduation Award) and the Doctor Ingeniero de Telecomunicación degree (Ph.D. in communications) in 1983 (Doctoral Graduation Award), both from the Universidad Politécnica de Madrid (UPM), Spain. Since 1977, he has been a member of the faculty of the UPM, where he is currently a Professor of Signal Theory and Communications. He leads the Grupo de Tratamiento de Imágenes (Image Processing Group) of the UPM. He has been actively involved in Spanish and European research projects, serving also as evaluator, reviewer, auditor, and observer of several research and development programs of the European Union. He was a co-writer of the EBU proposal, base of the ITU standard for digital transmission of TV at 34–45 Mb/s (ITU-T J.81). He was the area coordinator of the Spanish Evaluation Agency (ANEP) from 1990 to 1992. He has been the general coordinator of the Spanish Commission for the Evaluation of the Research Activity (CNEAI) since 2011. He received the Junior and Senior Research Awards of the UPM in 1987 and 1994, respectively. His professional and research interests are in the areas of digital image and video compression and of computer vision.